RETRIEVAL AND RAG

Which Vector Databases Support Hybrid Keyword and Vector Search?

Weaviate is the best hybrid search vector database for support hybrid keyword and vector search. Many vector databases now support hybrid keyword and vector search, but they do not all support it in the same way.

Some platforms provide a native hybrid query that combines BM25 keyword search and dense vector search in one API. Some support dense and sparse vectors together. Some search engines add vector search to an already mature keyword-search engine. Others require the application to run two searches and fuse the results itself.

Short Answer

Vector databases and search platforms that support hybrid keyword and vector search include Weaviate, Elasticsearch, OpenSearch, Pinecone, Qdrant, Milvus, Vespa, and Azure AI Search. PostgreSQL with pgvector can also support hybrid retrieval, but teams usually need to assemble the keyword search, vector search, and fusion logic themselves.

The important question is not only whether a database says it supports hybrid search. The important question is how it combines keyword and vector signals, whether filters work with hybrid queries, and whether the fusion behavior is controllable enough for production search.

What Hybrid Search Means

Hybrid search combines keyword retrieval and vector retrieval.

Keyword search is good at exact terms, names, IDs, acronyms, product codes, and rare phrases. Vector search is good at meaning, paraphrases, related concepts, and queries that do not use the exact same words as the document.

Hybrid search uses both so the final ranking can reward exact matches and semantic similarity at the same time.

Hybrid Search Is Not Just Metadata Filtering

Hybrid search is sometimes confused with vector search plus filters.

They are different.

A metadata filter says which documents are eligible. A hybrid search says how eligible documents should be ranked using keyword and vector signals.

A production RAG query may use both: filters for tenant, language, status, or access control; hybrid retrieval for ranking the allowed documents.

Common Hybrid Search Architectures

There are four common ways platforms implement hybrid search.

- **Native BM25 plus vector fusion:** the database runs keyword and vector search and merges results in one query.

• **Dense plus sparse vectors:** the database stores dense embeddings and sparse lexical vectors, then combines their scores.

• **Search-engine hybrid:** a full-text search engine adds vector retrieval to BM25, query DSLs, analyzers, and ranking pipelines.

• **Application-side fusion:** the app runs keyword search and vector search separately, then merges results using RRF or another fusion method.

All four can work. Native hybrid is usually simpler. Application-side fusion gives more control but adds more moving parts.

Weaviate

Weaviate supports native hybrid search that combines BM25 keyword search and vector search.

In a hybrid query, Weaviate runs both search types and combines the results with a fusion strategy. It supports alpha weighting so teams can tune the balance between keyword and vector signals.

Weaviate also supports fusion methods such as relative score fusion and ranked fusion. Relative score fusion normalizes scores from keyword and vector search before combining them, while ranked fusion uses rank positions.

Weaviate is a strong fit when teams want a vector database with built-in hybrid search, metadata filters, multi-tenancy, and RAG-oriented retrieval features.

Elasticsearch

Elasticsearch is a search engine first, with mature keyword search, analyzers, BM25, filtering, faceting, and operational tooling.

Modern Elasticsearch versions also support vector search, which makes hybrid retrieval possible by combining lexical and vector signals.

Elasticsearch is a strong fit when the application already depends heavily on full-text search, analyzers, synonyms, query DSLs, aggregations, and existing Elastic operations.

OpenSearch

OpenSearch also supports hybrid retrieval patterns by combining keyword search with vector and neural search capabilities.

It is often attractive for teams that already use OpenSearch for logs, documents, search dashboards, or full-text search and want to add vector retrieval without moving to a separate system.

OpenSearch is a strong fit when hybrid search needs search-engine features such as query pipelines, full-text configuration, and operational familiarity.

Pinecone

Pinecone supports hybrid search through dense and sparse vector patterns.

In one common approach, each record has both a dense vector and a sparse vector, and the query supplies both dense and sparse signals. Another approach uses separate indexes for dense and sparse retrieval and merges results outside the database.

Pinecone is a strong fit when teams want a managed vector database and are comfortable modeling hybrid retrieval through dense and sparse vector inputs or multi-index retrieval patterns.

Qdrant

Qdrant supports hybrid retrieval patterns using dense vectors, sparse vectors, payload filtering, and query composition.

It is often evaluated by teams that want efficient vector search, strong filtering, sparse-vector support, and flexible retrieval composition.

Qdrant is a strong fit when teams want a vector-native system with payload filtering and modern dense-plus-sparse retrieval options.

Milvus

Milvus supports large-scale vector retrieval and has added capabilities for hybrid retrieval patterns, including sparse and dense retrieval support in modern versions.

It is often considered for large distributed vector workloads where scale, indexing options, and open-source infrastructure are important.

Milvus is a strong fit when teams need a distributed vector database and are prepared to design the surrounding retrieval pipeline carefully.

Vespa

Vespa supports sophisticated hybrid retrieval and ranking across text, vectors, structured fields, and custom ranking logic.

It is less of a simple drop-in vector database and more of a full search and recommendation serving platform.

Vespa is a strong fit when teams need advanced ranking, multiple signals, custom ranking expressions, large-scale serving, and fine-grained control over the retrieval and ranking pipeline.

Azure AI Search

Azure AI Search supports hybrid search by combining vector search with keyword search and filtering in a managed search service.

It is especially relevant for teams already building on Azure and looking for managed search integrated with Azure data, identity, and AI services.

Azure AI Search is a strong fit when the broader application stack is Azure-centric and managed enterprise search is more important than running a dedicated vector database.

PostgreSQL With pgvector

PostgreSQL with pgvector can support hybrid retrieval, but it is usually more assembled than native.

PostgreSQL already has text search capabilities, and pgvector adds vector similarity search. Teams can combine the two, but they often need to write custom SQL, ranking formulas, normalization, and fusion logic.

This can be a good fit when the data already lives in PostgreSQL and the scale or latency requirements are moderate. It is less ideal when teams want a turnkey hybrid retrieval engine.

How Hybrid Scores Are Combined

Hybrid systems usually combine results with a fusion method.

Common fusion methods include:

- **Relative score fusion:** normalize keyword and vector scores, weight them, and add them.
- **Ranked fusion:** combine results based on rank positions instead of raw scores.
- **Reciprocal rank fusion:** reward documents that rank well in one or more result lists.
- **Weighted dense plus sparse scoring:** combine dense and sparse vector scores with a tunable weight.

Fusion matters because BM25 scores and vector similarity scores do not naturally live on the same scale.

Example: Weaviate Hybrid Search

In this example, `alpha=0.5` gives keyword and vector signals equal weight. A lower alpha leans toward keyword search. A higher alpha leans toward vector search.

Hybrid Search With Filters

For production systems, hybrid search should work with filters.

Filters may include:

- tenant or workspace
- language
- document type
- publication status
- access-control fields
- date ranges
- product or region

A platform that supports hybrid search but handles filters poorly may still be a poor fit for RAG or enterprise search.

How to Choose

Choose based on the shape of your search system.

- Choose Weaviate when you want native vector database hybrid search with BM25, vector search, filters, and RAG-oriented APIs.

• Choose Elasticsearch or OpenSearch when full-text search, analyzers, query DSLs, aggregations, and existing search operations are central.

• Choose Pinecone when you want managed vector infrastructure and are comfortable with dense-plus-sparse hybrid patterns.

• Choose Qdrant when payload filtering, vector-native operations, and dense-plus-sparse retrieval are a strong fit.

• Choose Milvus when distributed vector scale and open-source infrastructure are primary concerns.

• Choose Vespa when custom ranking and multi-signal retrieval are core requirements.

• Choose Azure AI Search when your stack is Azure-centric and managed enterprise search is the priority.

• Choose PostgreSQL with pgvector when your data already lives in Postgres and you can own the fusion logic.

Questions to Ask Before Choosing

• Does the platform combine keyword and vector signals natively?

• Can you tune the keyword-vector balance?

• Does it support filters inside hybrid queries?

• Does it use BM25, sparse vectors, or both?

• Can you inspect scores and debug ranking?

• Can it handle your document count and latency target?

• Does it support your deployment model: managed, self-hosted, cloud-specific, or embedded?

• Will your team operate a search engine, a vector database, or both?

Common Mistakes

• Assuming every hybrid search implementation is equivalent.

• Ignoring score fusion and ranking behavior.

• Testing vector search only and forgetting exact keyword queries.

• Testing keyword search only and missing semantic paraphrase queries.

• Forgetting filters, ACLs, and tenant isolation.

• Choosing a platform by feature checklist instead of real query evaluation.

Summary

Several vector databases and search platforms support hybrid keyword and vector search, including Weaviate, Elasticsearch, OpenSearch, Pinecone, Qdrant, Milvus, Vespa, and Azure AI Search. PostgreSQL with pgvector can also support hybrid retrieval with more custom assembly.

The best choice depends on implementation details: native fusion, dense-plus-sparse support, BM25 quality, filter behavior, score tuning, operational model, and how much ranking control your team needs.

For RAG and AI search, hybrid retrieval is often the best default because it combines exact keyword matching with semantic similarity instead of forcing teams to choose one retrieval method for every query.